Predicting Dry Weight in Hemodialysis Patients Using Machine Learning

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Computer Science & Engineering

by

PEDDU PAVANI(MT23MCS012)

under the guidance of

Dr. Poonam Sharma

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

Predicting Dry Weight in Hemodialysis Patients Using Machine Learning

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Computer Science & Engineering

by

PEDDU PAVANI(MT22MCS012)

under the guidance of

Dr. Poonam Sharma

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Peddu Pavani**, hereby declare that this dissertation titled **" Predicting Dry Weight in Hemodialysis Patients Using Machine Learning "** is carried out by me in the Department of Computer Science & Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree at this or any other Institution.

P.pavani
Name and signature
Date: 19/06/25

Certificate

This to certify that the dissertation titled **"Predicting Dry Weight in Hemodialysis Patients Using Machine Learning "** is submitted by **Peddu Pavani** in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Dr. Poonam Sharma
Department of Computer Science &
Engineering, VNIT Nagpur

सह. प्राध्यापक
Associate Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept of Computer Science & Engg
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
V.N.I.T. NAGPUR 0010 (India)

Dr. M. P. Kurhekar
Head, Department of Computer Science
& Engineering, VNIT Nagpur
Date:

विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., Nagpur-440010. (India)

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Non-Plagiarism Certificate

Certified that M. Tech Dissertation titled **" Predicting Dry Weight in Hemodialysis Patients Using Machine Learning "** submitted by **Peddu Pavani**, Enrollment No. **MT23MCS012**, ID No. **30495** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.

pavani_thesis (2).pdf

ORIGINALITY REPORT

10%	7%	6%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

Peddu Pavani P. Pavani

Enrollment No: MT23MCS012

Dr. Poonam Sharma
Associate Professor
Department of Computer Science & Engineering
VNIT, Nagpur

सह. प्राध्यापक
Associate Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept of Computer Science & Engg
वि रा रा रा स नागपुर-10 (भारत)
VNIT NAGPUR 0010 (India)

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to all who have supported and guided me throughout my M.Tech journey.

My sincere thanks to my supervisor, **Dr. Poonam Sharma**, for her exemplary guidance, steadfast support, and insightful feedback, which have been fundamental to my academic growth.

I am grateful to **Dr. M. P. Kurhekar**, Head of Department, for his invaluable support and motivation throughout this project.

The faculty members of the Department of Computer Science & Engineering at Visvesvaraya National Institute of Technology Nagpur are also to be thanked for their assistance, guidance, and knowledge in securing the resources required for this research.

I would also like to express my gratitude to my Family and Friends for their unwavering support, understanding, and encouragement throughout this journey. Their love, motivation, and belief in my abilities have been the driving force behind my accomplishments.

I also offer my silent reverence to the One whose fierce grace watches over me, guiding me through shadows into light. Her presence needs no name—only devotion.

Thank you all.

Peddu Pavani

ABSTRACT

Hemodialysis is a medical treatment which is used for individuals whose kidneys have failed or who can no longer effectively filter waste products, toxins from the blood, and excess fluids. This procedure works by removing the patient's blood and passing it through a dialyzer, a machine that functions as an artificial kidney. Inside the dialyzer, the semi-permeable membrane allows waste substances like urea, creatinine, excess salts, and water to pass from the blood into a cleansing fluid called dialysate, and that cleaned blood is returned into the body. When a person is going through hemodialysis treatment, it is very important to guess their Dry Weight correctly. Making the wrong prediction can cause major health problems like low blood pressure, too much fluid, cramps, or even heart difficulties. Doctors experience and manual observations are the main ways that people have traditionally found Dry Weight. These methods may not always be correct. In this thesis, proposed different machine learning approaches for predicting the Dry Weight in hemodialysis patients utilizing clinical related and demographic related information.

At AIIMS Nagpur, the data for this research was collected. It has 517 recordss of dialysis sessions, including important information like blood pressure, weight before and after dialysis, fluid evacuated, and pulse rate. Handling missing data, scaling the features, utilizing DBSCAN to find outliers, and finding important features using correlation analysis are all procedures that come before the main analysis. There are many different types of machine learning models like multiple linear regression, decision trees, random forests, XGBoost, and support vector regressors (SVR). In addition, more complex methods are used, such as domain-based stacking and granular bell membership in SVR and the grey wolf optimization which is used to fine-tune the hyperparameters of the XGBoost model.The SVR with a granular bell function is the best of all the models since it is giving the minimum error and the highest R^2 score. Tested different models on real-time test situations to see how fair and consistent they were. We also made a web app that makes it easy for patients to give us information and get forecasts about their Dry Weight.

This thesis demonstrates that machine learning models can provide precise and quick solutions for estimating Dry Weight, there by assisting physicians and enhancing the dialysis process for patients.

TABLE OF CONTENTS

Title **Page No.**

ABSTRACT . . . . . **iv**

TABLE OF CONTENTS . . . . . **v**

LIST OF TABLES . . . . . **vii**

LIST OF FIGURES . . . . . **viii**

1 INTRODUCTION **1**

1.1 Background and motivation 1

1.2 Problem statement 2

1.3 Objectives 3

1.4 Key contributions of the thesis 3

1.5 Thesis agenda 4

2 LITERATURE SURVEY **5**

2.1 Predicting dialysis adequacy with machine learning techniques 7

2.2 Using artificial intelligence to optimize anemia management in end-stage renal disease 7

2.3 Artificial intelligence for clinical decision support optimizes anemia management in hemodialysis patients 8

2.4 Using machine learning to predict changes in dry weight among hemodialysis patients 9

3 IMPLEMENTATION **12**

3.1 Dataset 12
3.1.1 Handling missing values in numerical columns 14
3.1.2 Handling categorical columns 14
3.1.3 Data normalization 15
3.1.4 Correlation for finding important features 15
3.1.5 Removing outliers 16
3.1.6 Data splitting 17
3.2 Methodologies 17
3.2.1 Multiple linear regression with interaction terms 17
3.2.2 Decision tree using fair split 22
3.2.3 Domain based random forest using staking 26
3.2.4 Support vector regressor using granular clustering 29
3.2.5 XGBoost using grey wolf optimization 32
3.3 Performance evaluation metrics 35
3.4 Real time test cases 36

4 EXPERIMENTS AND RESULTS 38
4.1 Real time Test cases 41
4.1.1 Multiple linear regression with interaction terms 41
4.1.2 Updated decision tree 42
4.1.3 Domain based random forest with staking 43
4.1.4 SVR with granular bells 44
4.1.5 XGBoost with grey wolf optimization 45
4.2 Graphs: Taylor plots 46

5 CONCLUSION AND FUTURE WORK 53
5.1 Conclusion 53
5.2 Future Work 54

List of Tables

3.2 Regression performence mtrics 35
3.3 Regression performence Metrics 36
4.1 All models performance metrics 38

List of Figures

3.1 DBSCAN 16

3.2 Decision Tree using fair split flow chart 25

3.3 Domain-based random forest using stacking flow chart 28

3.4 SVR training with granular clustering refinement flow chart 31

3.5 XGBoost using grey wolf optimization flow chart 34

4.1 Taylor diagrams for different models 46

4.2 SVR model performance for different number of features 49

4.3 SVR model performance for different number of features 50

Chapter 1

INTRODUCTION

1.1 Background and motivation

Hemodialysis is a medical treatment which can be used for individuals who have kidney failure or whos kidneys are no longer effectively filter waste products, toxins from the blood, and excess fluids. In this process, blood is taken from the patient and filtered through a machine known as a dialyzer. Inside the dialyzer, the biological membrane which permit unwanted substances like urea, creatinine, excess salts, as well as water to pass from the blood into a cleansing fluid named as dialysate, and that cleaned blood is returned into the body. Most of the time, hemodialysis is done in a week two or three times, and each session will take three to five hours, depending on how sick the patient is. It helps keep the body's electrolytes in balance, controls blood pressure, and manages fluid levels. This treatment is necessary for individuals with end stage kidney disease, kidney function is significantly reduced but not yet at complete failure, or specific instances of acute kidney damage.

Some of the most crucial aspects of dialysis treatments are preserving the proper amount of fluid. People with compromised renal function cannot normally get rid of extra fluid, so water, salt, as well as additional substances accumulate in the body. Left unremoved, liquid overflow can cause lung damage, heart failure from congestive hypertension, and an elevated fatality rate. Therefore, to preserve patients and prevent both immediate and lasting problems, it is imperative to exactly determine the optimal quantity of fluid to

be removed after every dialysis meeting. The result represents the idea of "Dry Weight," defined as the weight of the individual after dialysis treatments while all the extra fluid has been fully eliminated. The person who needs treatment is usually believed to be in a euvolemic state—this means not overhydrated and not dehydrated either. In clinical settings Dry Weight is used to decide how much fluid to remove in each dialysis session, a process called ultrafiltration. But finding the right Dry Weight is not straightforward at all; it is quite hard and also very personal. Many things can affect it, like changes in body shape, inflammation, food intake, or other health problems. If you guess the Dry Weight wrong, two things can happen. Too much fluid stays in the body, and that may cause heart issues, or too much fluid is removed, which leads to low BP, muscle pains, tiredness, and even damage due to less oxygen.

That's why it is very important to assess and update the Dry Weight again and again, because fluid balance plays a big role in how long a patient survives and their disability risk. In the past, this kind of checkup mostly depended on doctors' experience, what patients said, and trying different changes—but all these can be random and have chances of mistake. To get better accuracy in predicting Dry Weight in hospitals, we need to move towards using more data-driven, clear, and personalized methods such as those offered by machine learning.

1.2 Problem statement

In hemodialysis treatment prediction of a patient's accurate Dry Weight is important for effective fluid removal and maintaining Low blood pressure (hypotension). Incorrect prediction can lead to severe complications such as intradialytic hypotension, fluid overload, and long-term cardiovascular issues. Traditionally, practitioners often use subjective clinical judgment to assess Dry Weight. With the expanding availability of the patient health records along with the dialysis session data and machine learning offers a promising solution for developing a more objective and consistent method for Dry Weight prediction. The main goal of this thesis is to use machine learning models to predict the Dry Weight of hemodialysis patients based on important clinical, demographic, and treatment-related factors.

1.3 Objectives

- Implement different machine learning models to predict the target body weight (Dry Weight) of hemodialysis patients and find the one that gives the best predictions based on the evaluation metrics.
- To create a web app that is easy for healthcare providers to use and lets them enter patient information like weight, age, and blood pressure and get the predicted Dry Weight back, which will help them to make clinical decisions.

1.4 Key contributions of the thesis

This thesis describes different machine learning model that were implemented to predict the Dry Weight of individuals undergoing hemodialysis treatment. It makes use of both medical records and personal information about patients. Here, a preprocessing pipeline was used, such as DBSCAN to find the noise data points and Z-score or Min-Max normalization to make all the numbers look like they were on the same scale. Here Important features are selected by evaluating which ones correlate well with the target feature. We implemented different machine learning methods, such as SVR, XGBoost, random forest, decision tree and multiple linear regression. We used the RMSE, MAE, and R2 scores to see how well these models were performing. SVR with a granular bell membership function gives the minimum error and maximum R^2. XGBoost by using grey wolf optimization, which gives the better performance after SVR. To make sure the models are strong and fair and implemented real-world test cases such as giving the same input over and over, giving zero values, or changing the gender to see if the model is biased. We also created a website that allow the medical staff to quickly input patient information and receive a Dry Weight prediction. The website demonstrates the utility of this system in actual hospitals. In general, this thesis shows that machine learning is a better, more fair, and more accurate way to predict Dry Weight than the old methods, which mostly rely on the doctor's guess.

1.5 Thesis agenda

- Chapter 1: Introduction

  This chapter defines the problem definition, motivation behind it, objectives, and key contributions of this thesis.

- Chapter 2: Literature Survey

  In this chapter offers a review of existing work that guided the project work.

- Chapter 3 : Implementation

  This chapter contains the dataset description, preprocessing steps, and implementation of different optimized machine learning models.

- Chapter 4: Experiments and Results

  In this chapter includes the final results of different models and performance analysis of all the different models.

- Chapter 5: Conclusion and Future Scope

  This chapter provides the conclusion drawn from the research and ideas for further research.

Chapter 2

LITERATURE SURVEY

Accurate prediction of Dry Weight (DW) in patients who are undergoing hemodialysis (HD) is critical for managing fluids and getting good long term results. Dry Weight is the least weight of a patient can have after dialysis without having signs of being too hydrated. The difficulties in determining Dry Weight could lead to serious health issues, including intradialytic hypotension, left ventricular enlargement, and increased death rates [1]. Because of this, the main goals of kidney disease research is to make Dry Weight estimation more accurate. The procedure is supposed to reduce the chance of heart problems and make dialysis work better. In the past, doctors mostly relied on their judgment to figure out Dry Weight. This included physical exams, keeping track of blood pressure trends, and looking at the patient-reported symptoms. Although these methods are inexpensive and non-invasive they still rely on subjective judgment. Controversies and problems of volume control and hypertension in hemodialysis [2]. This research says that when doctors use subjective methods (based on their own judgment), the results can be different for each doctor. This inconsistency can make it difficult to keep track of fluid levels in the body, which can cause heart problems over time.

A lot of dialysis centers still use these clinical (subjective) methods, but their flaws show that we need more reliable and standardized methods that are based on objective data (not just personal judgment). Recently, bioimpedance spectroscopy (BIS) has become a popular objective way to figure out Dry Weight. BIS measures the electrical conductivity of body tissues to tell the difference between the intracellular and extracellular fluid com-

partments, the studies on using bioimpedance to manage fluid in patients on maintenance hemodialysis showed that BIS-guided dialysis can greatly lower excess fluid and improve heart health measures like the size of the left ventricle. These results indicate that BIS offers both more precise assessment of DW and also enhances clinical outcomes.

Other newly developed non-surgical methods like lung ultrasound (LUS) and measuring the diameter of the posterior vena cava (IVC), have been looked into for finding Dry Weight. LUS can find B lines, which show that the lungs are congested. It has also been helpful as an extra tool for checking volume status. Chest ultrasound for identifying pulmonary congestion in patients undergoing dialysis [5] discovered that the incorporation of LUS into clinical practice enhanced the accuracy of Dry Weight assessment, especially when combined with traditional techniques. Furthermore, echocardiographic measurements of IVC diameter have demonstrated potential as indicators of central venous pressure and volume overload, as explained by Agarwal et al. [6], this method's accuracy largely depends on respiratory fluctuations and the operator's expertise

Recent developments in artificial intelligence and machine learning have yielded innovative methods for forecasting Dry Weight.Trained machine learning models like random forests and support vector machines (SVM) on multi-dimensional data from dialysis sessions, clinical variables, and comorbidities. Kim et al. [7] showed that these models performs great at finding fluid imbalances than traditional linear regression methods because They demonstrated greater ability to correctly identify both detected and non-detected cases. These methods are promising, however they require substantial computing power and large labeled datasets, which can be a problem when trying to use them in real life. With the advancement of Modern health-monitoring technologies like smart sensors and IoT frameworks, it is now possible to monitor patients in real time. Researchers are developing devices capable of continuously measuring parameters such as bioimpedance, heart rate variability, and blood pressure. These smart devices when they will integrated with artificial intelligence have the potential to dynamically manage Dry Weight, adapting to patients' changing needs over time. Kooman et al. [8] noted that such integration facilitates individualized dialysis and the development of adaptive, real time fluid control systems that potentially transforming conventional hemodialysis practices.

2.1 Predicting dialysis adequacy with machine learning techniques

Studied the implementation of predictive machine learning (ML)[30] and deep learning models to evaluate the adequacy of dialysis treatments specifically focusing on the Urea Reduction Ratio (URR). The Urea Reduction Ratio (URR) It shows the percentage decrease in Blood Urea Nitrogen (BUN) levels before and after a dialysis session. Previously, researchers assessed the appropriateness of dialysis treatments using URR and Kt/V, which frequently involved invasive blood tests and could be costly and disruptive. This research aimed to develop non-invasive forecasting techniques using routinely acquired data from a hemodialysis treatment.

The study participants looked at data from 1,333 dialysis treatment exercises encompassing 61 individuals. Every experiment generated time series data on dialysate flow rate, ultrafiltration volume, and blood flow. Among the many kinds of models examined were two DL architecture like convolutional neural network architectures (CNN) and gated recurrent units (GRU), linear regression, random forest, and XGBoost.

The XGBoost showed among these the best with a MAPE of 2.500, RMSE of 2.906 and Spearman correlation of 0.873. Both CNN and GRU also did reasonably well, outperforming regression-based models. Particularly shown to be the most important indicator of URR was pre-dialysis weight alone.

2.2 Using artificial intelligence to optimize anemia management in end-stage renal disease

Artificial intelligence (AI) models in ameliorating the therapy of anemia in patients with end-stage renal disease (ESRD)[32]. Erythropoietin-Stimulating drugs have been utilized to treat anemia in individuals with chronic renal disease. Historically, the selection and dosing of these ESAs have depended on broad trial-and-error methods rather than individualized therapy procedures; these medications have side effects that highlight how crucial it is to customize treatment routines for each patient. In this research dialysis

centers tested the Anemia Control Model (ACM), a type of artificial intelligence model on 752 patients. The model which used artificial neural networks, figured out the best ESA doses by looking at the patient's hemoglobin levels, previous doses, and other factors.

A 6% increase in hemoglobin levels outside of the ideal range (10–12 g/dl) was a big deal. The drop was 12.6% of people who ACM told to strictly follow. Along with a drop in blood transfusion rates the company's dosage was found to have reduced by 27%. The study indicates that personalized dosing of ESA based on AI can reduce the hemoglobin variability and cut down on the use of erythropoietin stimulating agents (ESAs). Artificial intelligence models adapt to the response of each patient enhancing safety and efficacy compared to conventional fixed rule dosing. Their findings suggest that to effectively manage bleeding, these advanced technologies should be included in regular kidney treatment and tested more thoroughly in large clinical trials.

2.3 Artificial intelligence for clinical decision support optimizes anemia management in hemodialysis patients

An international observational study to evaluate the influence of the Anemia Control Model (ACM)[32], an artificial intelligence (AI) clinical decision support system, on the management of anemia in patients undergoing hemodialysis. The study included 752 patients from three NephroCare clinics in different countries and was done in two parts the first part. A thelve month control phase uses conventional anemia management followed by a subsequent twelve month observation phase regulating the dosing of the erythropoiesis-stimulating agent (ESA) by the ACM.

After the introduction of ACMs, there was a big reduction in ESA dosage, as seen by the drop in median darbepoetin use from 0.63 to 0.46 μg/kg/month. After following ACM recommendations, the proportion of hemoglobin readings that fell within the target range increased from 70.6% to 76.6%, reaching 83.2%. There was a reduction in the variation of hemoglobin levels among patients, indicating better control of anemia. These results show that AI-assisted decision support can improve hemodialysis anemia management in dialysis patients, leading to better clinical outcomes and possibly lower costs.

2.4 Using machine learning to predict changes in dry weight among hemodialysis patients

The study aimed to develop machine learning models to predict the need for adjustments to dry weight (DW) [39] in hemodialysis patients. Dry weight or The target post-dialysis weight without any excess fluid, is particularly critical for attaining effective results for patients. However determining Dry Weight often depends on the physicians judgment. which can be different for each person and isn't always right. This paper suggests an objective, data-driven method that uses machine learning to improve these evaluations.

The research employed data from 69,375 dialysis sessions involving 314 patients at a single hospital in Japan. The researchers implemented two Random Forest models: one to find out when DW should go up and another to find out when it should go down. The models used over 150 different features of information, like changes in blood pressure and heart rate during dialysis, weight gain, C-reactive protein (CRP), albumin, and other medical data. They data split into 75% for training and 25% for testing. Then they used SMOTE method to make the dataset even. The models did well, with the DW increase model getting an AUC of approximately 0.70 and the DW decrease model getting an AUC of around 0.74. The most important elements that caused dry weight to go up were declines in blood pressure at the time of dialysis. High CRP and low albumin levels were crucial signals that DW was going down. They showed symptoms of inflammation and poor nutrition, these results were consistent with clinical guidelines and indicated that the models could facilitate real time evaluation.

The study included both males and females: 314 patients total but did not Conduct separate analyses for male and female participants, therefore, there was no model trained separately on males only, nor was sex specifically analyzed for its impact on the performance of the model.

Calculation of dry weight in hemodialysis patients

Standard clinical method

A trial and error procedure based on clinical observation is the most typical way that dialysis centers work [38]. First, an approximate dry weight is chosen. Then, throughout each dialysis session, fluid is slowly taken out. After dialysis, the patient is checked for symptoms like low blood pressure, dizziness, or cramps (which could mean too much fluid has been removed) or for indicators like swelling and high blood pressure (which could mean too much fluid has been added). The objective for fluid removal is changed over several sessions until the patient feels good, has no symptoms, and has stable blood pressure after dialysis. This method is easy to use and doesn't need any specific tools, but it is quite subjective and can change from one doctor to another.

Body composition monitoring (BCM) / Bioimpedance spectroscopy (BIS)

Bioimpedance analysis is a more advanced and objective way to figure the dry weight. By transmitting low-level electrical currents through the body, it measures resistance and reactance, which measures fluid levels in different parts of the body. The device figures out how much water is in the body (TBW), outside the cells (ECW), within the cells (ICW), and most crucially, how much extra fluid there is (overhydration, or OH). To find the dry weight, you take the present body weight and subtract the amount of water that is too much. This procedure is faster, and more accurate than a clinical assessment, but it needs special tools and skilled staff.

Monitoring relative plasma volume (RPV)

Another way to check the concentration of plasma as fluid is withdrawn during dialysis is to utilize RPV monitoring. A quick drop in plasma volume could mean that the patient is being dehydrated, whereas a sluggish or flat curve could mean that they are still getting too much fluid. This feedback in real time helps doctors figure out if too much or too little fluid is being taken out throughout the session. But this method requires a specific monitor

and can be influenced by factors such as the patient's position and blood pressure.

Ultrasound and echocardiography of the inferior vena cava (IVC)

In more advanced or hospital settings, ultrasonography can also be used to measure the inferior vena cava (IVC), which is the major vein that brings blood back to the heart. An IVC that is dilated and doesn't collapse shows that there is too much fluid, while an IVC that is small and collapses shows that there is not enough fluid. Echocardiography can also check fluid levels by looking at how well the heart works and how much pressure it can hold. These imaging techniques yield significant objective data, however they necessitate access to specialized equipment and trained personnel, hence constraining their normal application in standard dialysis facilities.

Chapter 3

3.1 Dataset

and CheLaxpan oversaw the collection of the dataset from the AIIMS Nagpur
. Department...The All India Institute of Medical Sciences Nagpur (AIIMS

Dr. An
Nephrolog

Feature description

Feature Name	

Feature Name	Description
IntradialyticSBP3	Systolic blood pressure during dialysis (third reading).
IntradialyticDBP3	Diastolic blood pressure during dialysis (third reading).
SystolicBP postDialysis	Systolic blood pressure post dialysis.
DiastolicBP postDialysis	Diastolic blood pressure post dialysis.
Dry Weight	Clinically estimated Dry Weight of the patient.

3.1.1 Handling missing values in numerical columns

It is common in datasets to have missing values for numerical columns. If you do not handle them, they can lead to biased results or errors at the time of training a model. These gaps are filled in using different imputation methods that keep the overall structure and distribution of the data.

Mean imputation is one of the most common ways to deal with missing numbers. This method fills in the values missing by averaging of the non missing entries in the same column. This method works best when the data is evenly spread out and doesn't have any big outliers. It is also quick and easy to use.

3.1.2 Handling categorical columns

When categorical columns contain missing values, it can lead to poor data quality and impact model accuracy. Imputation methods used fill in missing entries correctly because categorical data unlike numbers have distinct labels.Mode imputation is one of the easiest and most common methods. It fills in missing values in the categorical column with the category that happens most repeatedly. When one category dominates the column and stays pretty stable, this method works well.

Categorical data includes variables that stand for categories or labels but don't have any inherent numerical meaning (like gender, blood type, or color). Most of the machine learning algorithms on the other hand and need numbers as input. One hot encoding is one of the most common ways to change categorical features into a format that can be used by ML algorithms. One hot encoding changes each categorical value into a binary vector,

with each new category becoming a new column. In these new columns, the column that matches the category gets a 1 and all the other columns get 0s.

3.1.3 Data normalization

Transforming data is an important important step in data preprocessing. It makes sure that the numerical characteristics are put on a similar scale before they are used in a machine learning model.

Min-Max normalization:

Typically spanning [0, 1], min-max normalization grows used to the data at a set frequency. When the data characteristics included maximum and minimum values, min-max normalization became particularly helpful. One reaches this transition by the

$$X_{\text{scaled}} = \frac{X - X_{\text{minimum}}}{X_{\text{maximum}} - X_{\text{minimum}}} \tag{3.1}$$

where X represents the original value, and X_{minimum} and X_{maximum} are the minimum and maximum values of the feature, respectively. These modifications guarantee prevention of dominance by large values and improved model convergence by being done prior to data split into sets for testing and training.

3.1.4 Correlation for finding important features

Correlation is a means to see how closely two things are related to each other. Correlation is helpful for choosing features since it reveals how closely each feature is related to the target variable or how similar features are to each other.

Pearson correlation It indicates how strongly and in what direction two continuous variables are related in a straight line. The coefficient could be anywhere between -1 and +1.

+1: This means that there is a completely positive linear relationship.

0: This means there is no linear relationship.

-1: Shows a completely negative linear relationship

$$r = \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{n}(y_k - \bar{y})^2}} \tag{3.2}$$

Where:

- r: Pearson's coefficient represents the degree of linear correlation
- x_k, y_k: The kth value in dataset
- $\bar{x}, \bar{y}$: Average of x and y respectively
- n: Total count of data observations

3.1.5 Removing outliers

DBSCAN is a density-based grouping method that finds outliers by looking at their clusters with few data points and density in the attribute space . It finds various groups in the data space by looking for dense areas that are separated by low density areas. DBSCAN works well with real-world data problems, unlike K-Means or a hierarchy of clustering, which assumes that clusters are light and elliptical.In the DBSCAN plot, red dots represent clear outliers identified by the algorithm, while some isolated blue dots far from dense clusters may also be potential outliers.

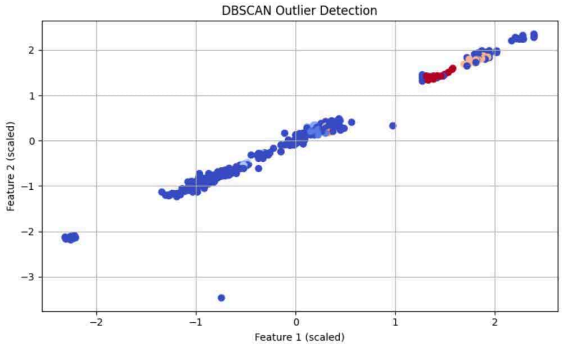


Figure 3.1: DBSCAN

3.1.6 Data splitting

Dividing the datasets is a key part of machine learning. This means dividing the data into smaller groups, such as sets used for training, validation, and testing. This process ensures effective model learning and accurate performance evaluation.

Training set: This set is used during the training of the model. We used 80% of the data to train.

Validation set: Tuning hyperparameters and reducing the chance of overfitting. We used 10% of the data for validation.

Test set: Used to test how well the model is working and we used ten percent of the data for the test.

3.2 Methodologies

3.2.1 Multiple linear regression with interaction terms

Multiple linear regression [10] shows how a dependent variable is associated to more than one independent variable. Adding interaction terms [24] helps to show how two or more predictors work together to affect the dependent variable in a way that can not be explained by their individual effects.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \cdots + \beta_p x_p \tag{3.3}$$

Where:

- β_0: Represent the constant or intercept of the model.
- $\beta_1, \beta_2, \ldots, \beta_p$: Represent are the weights assigned to each corresponding predictor variable.
- y: The dependent variable .
- $x_1, x_2, \ldots, x_p$: The independent variables.

- β_3: It captures the combined influence of x_1 and x_2, reflecting their interaction.

The interaction term x_1x_2 lets the slope of x_1 fluctuate based on the value of x_2, which makes it easier to find more complex correlations between input features.

Gradient descent is an optimization algorithm that works in steps. It starts with regression coefficients and guesses for the updates them one step at a time to make a cost function, like the Mean Squared Error (MSE), as small as possible. To make convergence and stability better advanced optimizers like Adam (Adaptive Moment Estimation) can be used especially when gradients are noisy. By keeping exponentially weighted moving averages of previous gradients and their squared values, Adam combines the best parts of momentum and RMSProp. The algorithm updates the regression coefficient vector.

Adam Optimizer Update Rule:

$$\beta_t = \beta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.4}$$

Where:

- β_t: It represents the updated model parameter at time t.
- β_{t-1}: The model parameter at the previous step.
- $\hat{m}_t$: Adjusted estimate of the first moment
- $\hat{v}_t$: Adjusted estimate of the second moment
- α: Step size or learning rate used for updates
- ϵ: Tiny value added to prevent division by zero

Huber loss function:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases} \tag{3.5}$$

Where:

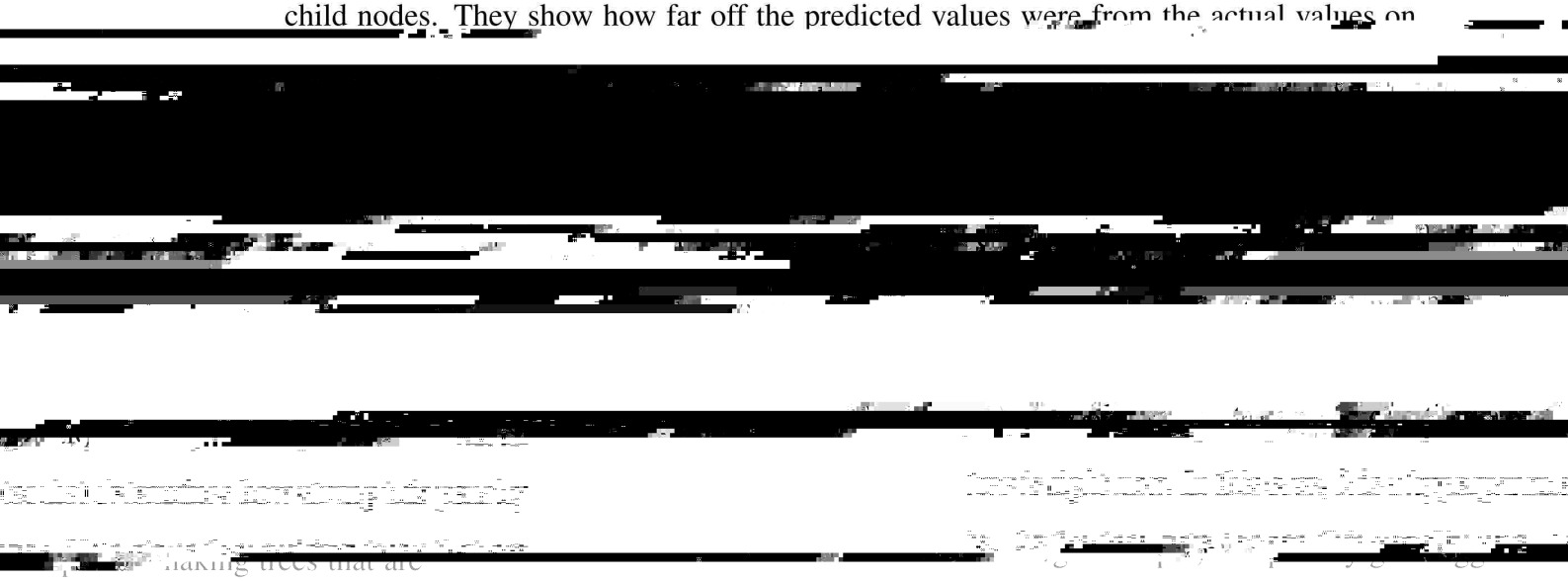
- $a = y_i - \hat{y}_i$: This represents the residual, which is the difference between the actual value (y_i) and the predicted value ($\hat{y}_i$) for each observation.
- δ: A user-defined threshold that determines the transition between quadratic and linear loss behavior

Solutions to multicollinearity

lasso regression .

3.2.2 Decision tree using fair split

After the split, MSE_L and MSE_R are the Mean Squared Errors of the left and right child nodes. They show how far off the predicted values were from the actual values on



del is supported to stay simple as
This makes it better at working

The hyperparameter γ determines
e the average predicted results for
e difference between these means
nodes. More the difference could
his makes the model more likely
ferent groups, it will prevent from
ions.

too complicated, which can lead to overfitting. This mo
well as more generalizable by punishing greater depth.
with data it hasn't seen before.

The term $\gamma \cdot |\bar{y}_L - \bar{y}_R|$ serves as a penalty for fairness.
how important this penalty is. The values of $\bar{y}_L$ and $\bar{y}_R$ a
the left and right child nodes respectively. The absolute
shows how different the predictions are between the two
mean that the split was unfair or biased, so punishing t
to make predictions that are fair and balanced across dif
overfitting ,this is especially important in sensitive situat

Algorithm 1 Decision tree regression with fairness aware splitting

Input: Dataset $\mathcal{S} = \{(\mathbf{z}_i, t_i)\}_{i=1}^{n}$
Parameters: max_depth, min_samples_split, min_samples_leaf, num_bins, λ, γ, α, level, penalty
function GROWTREE($\mathcal{S}$, level)
  if level $\geq$ max_depth **or** $|\mathcal{S}| <$ min_samples_split **then**
    return terminal node with output $\hat{t} = \frac{1}{|\mathcal{S}|} \sum t_i$ *(average target)*
  end if
  Compute MSE_{total}
  Initialize best_score $\leftarrow \infty$, best_split $\leftarrow$ None
  for all attribute a **do**
    Bin a into num_bins intervals
    for all threshold τ (midpoints of bins) **do**
      $\mathcal{S}_L \leftarrow \{(\mathbf{z}_i, t_i) \mid z_i[a] \leq \tau\}$
      $\mathcal{S}_R \leftarrow \{(\mathbf{z}_i, t_i) \mid z_i[a] > \tau\}$
      if $|\mathcal{S}_L| <$ min_samples_leaf **or** $|\mathcal{S}_R| <$ min_samples_leaf **then**
        continue
      end if
      Compute MSE_L and MSE_R
      Compute fairness_penalty: penalty $\leftarrow \gamma \cdot |\bar{t}_L - \bar{t}_R|$
      TotalScore $\leftarrow MSE_L + MSE_R - MSE_{\text{total}} + \lambda \cdot$ level $+$ penalty
      if TotalScore $<$ best_score **then**
        best_score $\leftarrow$ TotalScore
        best_split $\leftarrow (a, \tau)$
      end if
    end for
  end for
  if best_split is None **then**
    return leaf node with $\hat{t} = \frac{1}{|\mathcal{S}|} \sum t_i$
  end if
  Divide $\mathcal{S}$ into $\mathcal{S}_L, \mathcal{S}_R$ using best_split
  left_node $\leftarrow$ GROWTREE($\mathcal{S}_L$, level + 1)
  right_node $\leftarrow$ GROWTREE($\mathcal{S}_R$, level + 1)
  return internal node with best_split, left_node, right_node
end function
function MAKEPREDICTION(z, $\mathcal{T}$)
  current_node $\leftarrow$ root of $\mathcal{T}$
  while current_node is not a terminal node **do**
    if $z[a] \leq$ split value at current_node **then**
      current_node $\leftarrow$ left_childSS
    else
      current_node $\leftarrow$ right_child
    end if
  end while
  return prediction $\hat{t}$ at terminal node *(average of leaf)*
end function

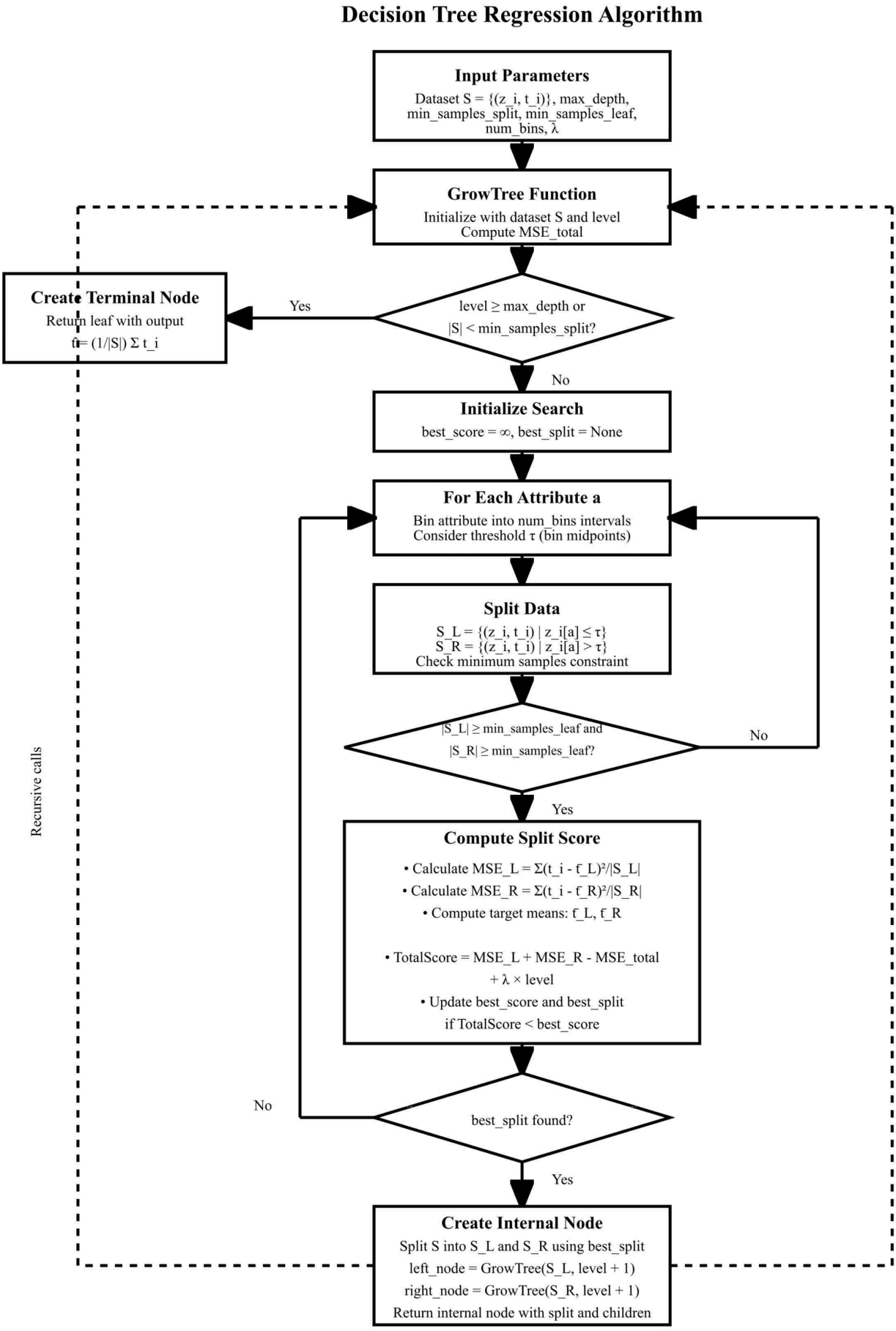

Decision Tree Regression Algorithm
Input Parameters
Dataset S = {(z_i, t_i)}, max_depth, min_samples_split, min_samples_leaf, num_bins, λ
GrowTree Function
Initialize with dataset S and level
Compute MSE_total
level ≥ max_depth or |S| < min_samples_split?
Yes
Create Terminal Node
Return leaf with output f̂ = (1/|S|) Σ t_i
No
Initialize Search
best_score = ∞, best_split = None
For Each Attribute a
Bin attribute into num_bins intervals
Consider threshold τ (bin midpoints)
Split Data
S_L = {(z_i, t_i) | z_i[a] ≤ τ}
S_R = {(z_i, t_i) | z_i[a] > τ}
Check minimum samples constraint
|S_L| ≥ min_samples_leaf and |S_R| ≥ min_samples_leaf?
No
Yes
Compute Split Score
• Calculate MSE_L = Σ(t_i - f̂_L)²/|S_L|
• Calculate MSE_R = Σ(t_i - f̂_R)²/|S_R|
• Compute target means: f̂_L, f̂_R
• TotalScore = MSE_L + MSE_R - MSE_total + λ × level
• Update best_score and best_split if TotalScore < best_score
No
best_split found?
Yes
Create Internal Node
Split S into S_L and S_R using best_split
left_node = GrowTree(S_L, level + 1)
right_node = GrowTree(S_R, level + 1)
Return internal node with split and children
Recursive calls

Figure 3.2: Decision Tree using fair split flow chart

3.2.3 Domain based random forest using staking

Random Forest for regression [11] is a machine learning technique that uses a more number of decision trees to predict a value. Each tree makes its own prediction, and the final answer is the average of all those predictions, it makes the predictions more reliable and less likely to be affected by noise or overfitting by using random samples of data and features for each tree.

Step 2: Train separate models for each group

For each domain, train a separate Random Forest model using the features of that domain:

- One model learns from vital signs.
- The other one solely looks at demographic data.
- The third one on the other hand is about clinical information.

Step 3:Get predictions from each model

Each of the three models is used to predict results using only the data it was trained on. So, the vitals model uses vitals data, the demographics model uses demographic data, and so on.

Step 4: Combine those predictions [26]

Next, take the predictions from all three models and put them together like new features. This combined data is then used to train another model (called a meta-model) that learns from the combined predictions to make even better decisions.

Step 5: Final prediction

Finally, we use the meta-model to give us the final result, based on what it learned from the three earlier models.

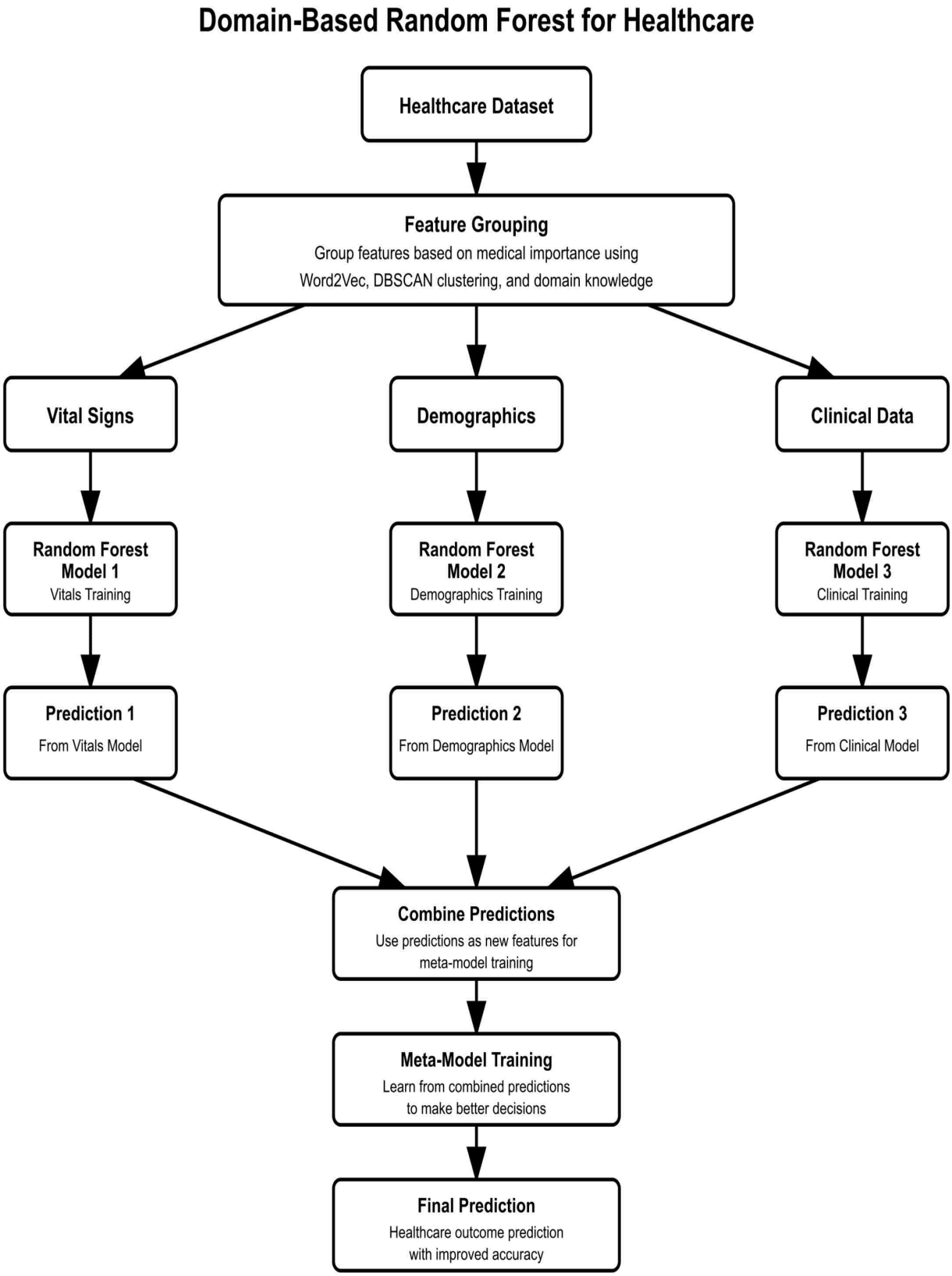


Figure 3.3: Domain-based random forest using stacking flow chart

3.2.4 Support vector regressor using granular clustering

Support vector machines (SVM) can be extended to regression tasks through a method known as Support vector regression (SVR) [27]. It targets to find a function that accurately predicts output values for given input values.

SVR can utilize either a linear or a non-linear kernel. The dot product of two input vectors makes up a linear kernel. A non-linear kernel is a more advanced function that can detect more complicated patterns in the data. The type of data and how demanding the job is will help you choose the right kernel.

We use something called Granular Bells to combine Support Vector Regression (SVR) and Granular bells [33] , this makes the model easier to understand and better at dealing with data that isn't clear or is noisy. We use DBSCAN, a method for clustering to put similar data points together. This allow SVR learn from these groups and make better predictions.

There are many benefits to using Granular Bells with SVM. Granular Bells help the SVM focus on local patterns within each group by breaking data into meaningful clusters or granules. This makes it better at handling data that is noisy, uncertain, or complex. This makes it easier to apply what you've learned to new situations and make more accurate predictions. Also, the bell-shaped membership functions make it easy to see how closely each data point belongs to a granule, which makes the results easier to understand and more reliable. Overall, adding Granular Bells to SVM makes the model more adaptable and dependable, which is important in real-world situations where data changes and noise are common.

Algorithm 2 SVR Training with Granular Clustering Refinement

Require: Feature set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, target values $\mathcal{Y} = \{y_i\}_{i=1}^{N}$
Initial DBSCAN parameters: ϵ_{init}, minPts
Refinement DBSCAN parameter: ϵ_{refine}
Error threshold: τ_{error}, max iterations: $T_{\max}$
Initial Clustering: Run DBSCAN on $\mathcal{X}$ with ϵ_{init}, minPts
for each cluster $\mathcal{C}_k$ **do**
Calculate cluster centroid: $\mathbf{m}_k \leftarrow \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \mathbf{x}_i$
Calculate average target: $\bar{y}_k \leftarrow \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_i \in \mathcal{C}_k} y_i$
end for
Initialize centroid set $\mathcal{M} \leftarrow \{\mathbf{m}_k\}$, target set $\mathcal{Y}_{\text{avg}} \leftarrow \{\bar{y}_k\}$
for $t = 1$ to $T_{\max}$ **do**
Train SVR model: $f^{(t)} \leftarrow \text{SVR.fit}(\mathcal{M}, \mathcal{Y}_{\text{avg}})$
Predict outputs: $\hat{y}_i \leftarrow f^{(t)}(\mathbf{x}_i)$ for all $i = 1, \ldots, N$
Calculate squared errors: $e_i \leftarrow (y_i - \hat{y}_i)^2$
for each cluster $\mathcal{C}_k$ **do**
Compute mean cluster error: $E_k \leftarrow \frac{1}{|\mathcal{C}_k|} \sum_{\mathbf{x}_i \in \mathcal{C}_k} e_i$
end for
Identify clusters exceeding error threshold: $\mathcal{S} = \{\mathcal{C}_k : E_k > \tau_{\text{error}}\}$
if $\mathcal{S}$ is empty **then**
break
end if
for each cluster $\mathcal{C}_s \in \mathcal{S}$ **do**
Re-cluster $\mathcal{C}_s$ using DBSCAN with ϵ_{refine}
Update centroids and target averages for refined sub-clusters
end for
Update $\mathcal{M}, \mathcal{Y}_{\text{avg}}$ with new centroids and targets
end for
return Final SVR model $f^{(t)}$, refined centroid set $\mathcal{M}$, and target averages $\mathcal{Y}_{\text{avg}}$

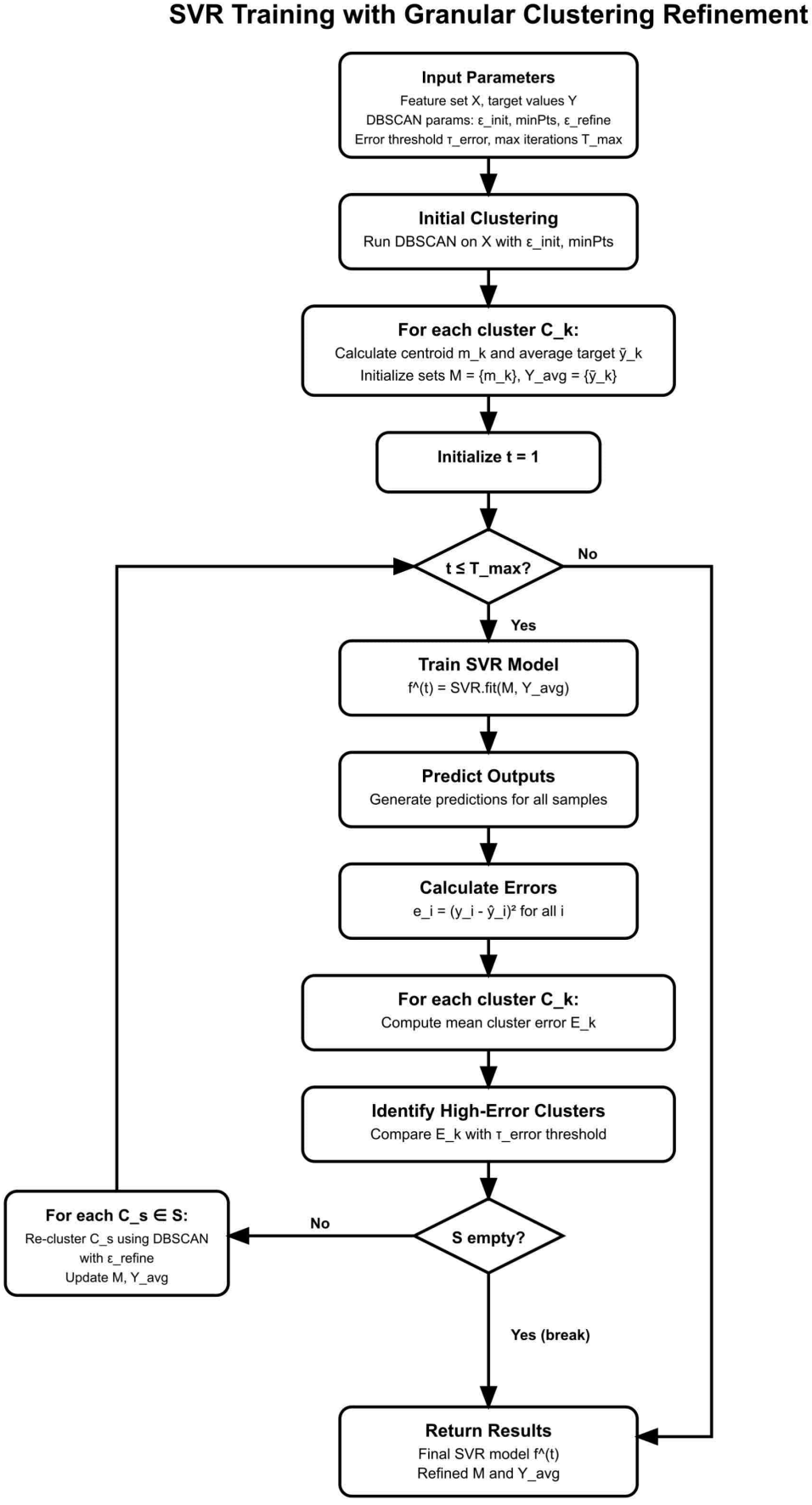


Figure 3.4: SVR training with granular clustering refinement flow chart

3.2.5 XGBoost using grey wolf optimization

XGBoost (Extreme Gradient Boosting) [37] is a quick and powerful machine learning technique that may be utilized for both regression and classification applications. It makes a set of decision trees, and each one tries to rectify the faults that the trees before it made. This is called boosting. XGBoost is different from other programs since it uses smart techniques like:

- Gradient Descent to reduce errors
- To stop overfitting use regularization
- Parallel computing to make computation go faster

The Grey Wolf Optimizer (GWO) [34] is a meta-heuristic algorithm that uses the natural way that grey wolves lead and hunt. Grey wolves are the top predators in their habitats and usually live and hunt in groups of 5 to 12 individuals. There is a set order of power in each pack. The alpha α wolves are the leaders of the pack and tell everyone else what to do. Beta β wolves are the alpha's advisors and are next in line to lead. Delta δ wolves can also work as scouts, hunters, or caregivers. The GWO algorithm is based on this social structure, with alpha, beta, and delta wolves leading the way and the other wolves following. This lets you strike a balance between exploration and exploitation throughout the optimization phase. Delta wolves can also have certain jobs, such as being scouts, hunters, or caregivers. The GWO algorithm is built on this social hierarchy, with the strongest solutions being led by alpha, beta, and delta wolves with the others following. This makes it possible to find a balance between exploration and exploitation during the optimization phase.

Algorithm 3 XGBoost hyperparameter optimization using grey wolf optimizer

Require: Dataset $\mathcal{D}$, population size N, maximum iterations T
Ensure: Optimized hyperparameter vector θ^*
Initialize population of candidate solutions $\{\theta_i\}_{i=1}^{N}$ randomly within the search space
Evaluate fitness (e.g., validation error) of each candidate θ_i by training XGBoost with θ_i
Determine leader solutions: alpha (θ_α), beta (θ_β), and delta (θ_δ) based on the best three fitness values
for $t = 1$ to T **do**
  Compute linearly decreasing parameter $a = 2 - \frac{2t}{T}$
  for each candidate θ_i **do**
    Generate random vectors $\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}(0, 1)$
    Compute coefficient vectors:

$$\mathbf{A}_k = 2a \cdot \mathbf{r}_1 - a$$
$$\mathbf{C}_k = 2 \cdot \mathbf{r}_2 \qquad \text{for } k \in \{1, 2, 3\}$$

    Update position based on leaders:

$$\mathbf{D}_\alpha = |\mathbf{C}_1 \odot \theta_\alpha - \theta_i|, \quad \mathbf{X}_1 = \theta_\alpha - \mathbf{A}_1 \odot \mathbf{D}_\alpha$$
$$\mathbf{D}_\beta = |\mathbf{C}_2 \odot \theta_\beta - \theta_i|, \quad \mathbf{X}_2 = \theta_\beta - \mathbf{A}_2 \odot \mathbf{D}_\beta$$
$$\mathbf{D}_\delta = |\mathbf{C}_3 \odot \theta_\delta - \theta_i|, \quad \mathbf{X}_3 = \theta_\delta - \mathbf{A}_3 \odot \mathbf{D}_\delta$$
$$\theta_i^{new} = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3}$$

    $\mathbf{D}$ is the distance between current solution and each leader
    $\mathbf{X}$ is the new position influenced by this distance and randomness; final position is the average of movements toward all three leaders
    Clip or round θ_i^{new} to ensure valid hyperparameter ranges/types
    Recalculate fitness by training XGBoost with updated θ_i^{new}
  end for
  Update leader candidates (θ_α, θ_β, θ_δ) according to latest fitness scores
end for
return θ_α as the optimal hyperparameter set

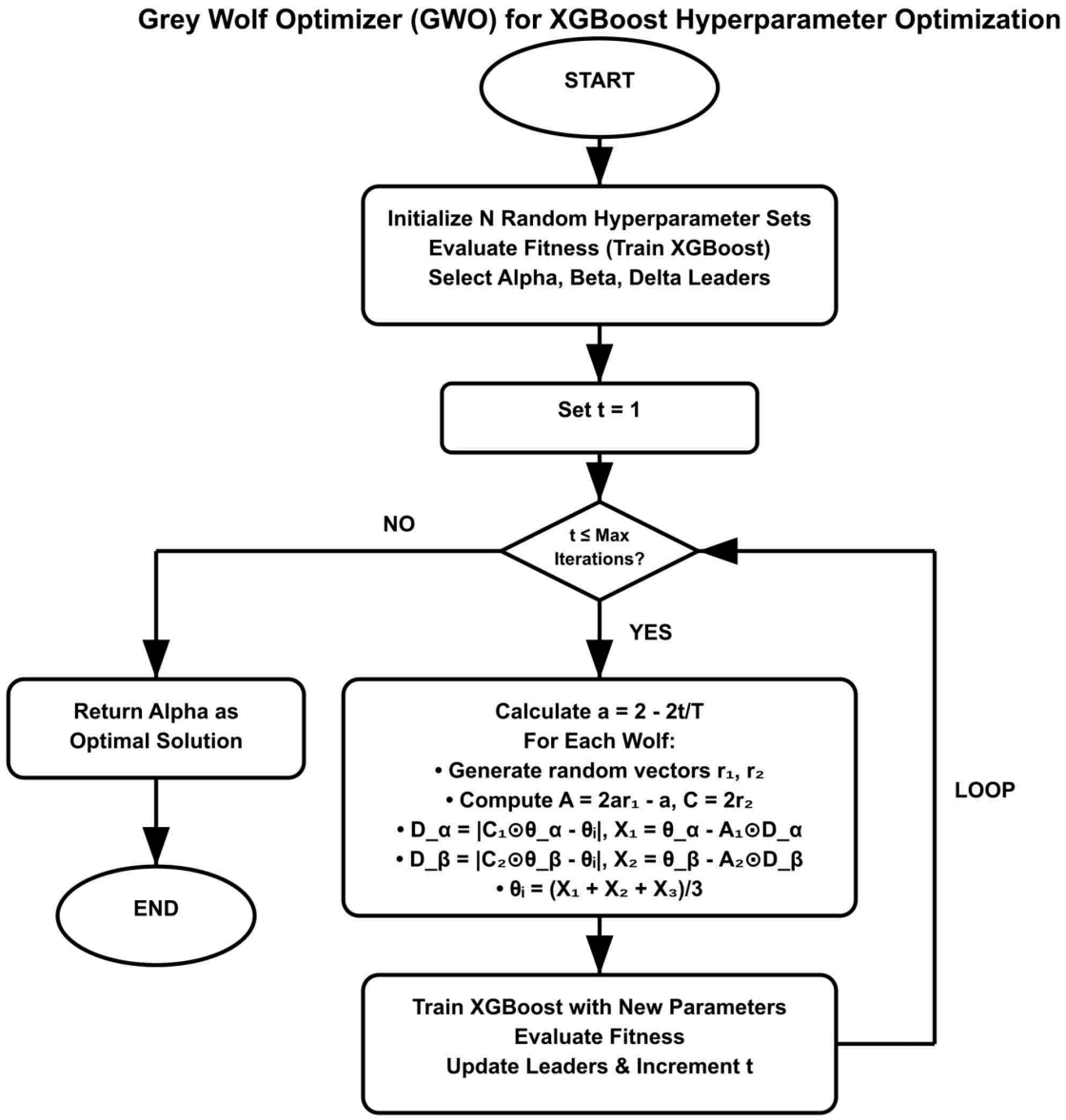

Grey Wolf Optimizer (GWO) for XGBoost Hyperparameter Optimization
START
Initialize N Random Hyperparameter Sets
Evaluate Fitness (Train XGBoost)
Select Alpha, Beta, Delta Leaders
Set t = 1
t ≤ Max Iterations?
NO
YES
Return Alpha as Optimal Solution
END
Calculate a = 2 - 2t/T
For Each Wolf:
• Generate random vectors r₁, r₂
• Compute A = 2ar₁ - a, C = 2r₂
• D_α = |C₁⊙θ_α - θᵢ|, X₁ = θ_α - A₁⊙D_α
• D_β = |C₂⊙θ_β - θᵢ|, X₂ = θ_β - A₂⊙D_β
• θᵢ = (X₁ + X₂ + X₃)/3
Train XGBoost with New Parameters
Evaluate Fitness
Update Leaders & Increment t
LOOP

Figure 3.5: XGBoost using grey wolf optimization flow chart

3.3 Performance evaluation metrics

Metric	Definition	Formula	Use of metric
MAE	Mean of absolute errors.	$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Calculate average estimation of error magnitude; it is useful in understanding typical size of error.
MSE	Mean of squared errors.	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Penalizes bigger errors heavily than MAE; Largly used in regression model error calculation.
RMSE	Root of mean squared errors.	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	gives the error in same units in the target; easier way of interpret in practical terms.
R^2	Variance explained by the model.	$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$	represents the amount of variance in the target is explained by the model.
Adj. R^2	R^2 adjusted for number of predictors.	$\text{Adj. } R^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$	Adjusts R^2 used for model complexity understanding;It helps compare the different models with different numbers of predictors.
MAPE	Mean absolute percentage error.	$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$	Expresses error in terms of percentage; useful for compare the performance of different models across scales.
SMAPE	Symmetric version of MAPE.	$\text{SMAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{(\lvert y_i\rvert + \lvert\hat{y}_i\rvert)/2}.$	Decrease the bias when actual values are nearest to the zero; better for datasets contains small or big values.
Median AE	Median of absolute errors.	$\text{MedAE} = \text{median}(\lvert y_1 - \hat{y}_1\rvert, \ldots, \lvert y_n - \hat{y}_n\rvert)$	Robust to the outliers management; useful when the data had been extremed values that had been skew mean-based metrics.
MBE	Mean bias error.	$\text{MBE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$	It uses to measure systematic bias; helping to detect over- or under-prediction.

Table 3.2: Regression performence mtrics

Metric	Definition	Formula	Use of metric
EVS	Explained variance score.	$\text{EVS} = 1 - \frac{\text{Var}(y_i - \hat{y}_i)}{\text{Var}(y)}$	Shows at the amount of proportion of variance captured by model; similar to that of R^2.
AIC	Akaike Information Criterion.	$\text{AIC} = 2k - 2\ln(L)$	model fit it will Balances and the complexity of model; lower AIC represents a better model.
BIC	Bayesian Information Criterion.	$\text{BIC} = k\ln(n) - 2\ln(\hat{L})$	Same like to AIC but add on to the a stronger penalty for complexity; suitable for all the types of small datasets.
RAE	Relative absolute error.	$\text{RAE} = \frac{\sum \lvert y_i - \hat{y}_i \rvert}{\sum \lvert y_i - \bar{y} \rvert}$	total error of model Compares to a naive error (mean model); lower values are always better it shows.
RSE	Relative squared error.	$\text{RSE} = \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$	Compares model squared error to data variance; similar to inverse of R^2.
Training Time	Time taken to train the model.	$T_{\text{train}} = \text{end}_{\text{train}} - \text{start}_{\text{train}}$	it represents the amount of time taken for training; large or frequently used its useful for updated models.
Testing Time	Time to evaluate on test data.	$T_{\text{test}} = \text{end}_{\text{test}} - \text{start}_{\text{test}}$	amount of time taken for testing (e.g., which operates on real-time systems).
Model Size	Memory usage of model.	$S_{\text{model}} = \text{Total Params} \times \text{Precision}$	represents deployment of the suitability, especially on the resource-constrained devices.

Table 3.3: Regression performence Metrics

3.4 Real time test cases

- **Test case 1: Consistency test of repeated input**

  Description: The average same input sample is given to the model 1000 times at the time of training and to check the stable real-world scenario.

  Purpose: This test ensures that the model produces reliable predictions when given identical input repeatedly.

  Use: It evaluates the determinism and mathematical stability of the model under perfectly controlled conditions.

- **Test case 2 : All zero input test**

  Description: Every input feature is set to zero.

Purpose: To observe how the model behaves in the absence of informative input.

Use: This test case used find the model's robustness to missing or zero value data. It helps identify how the model reacts when input data is incomplete.

- **Test case 3 : Maximum value test**

  Description: All input features are set to maximum possible values.

  Purpose: To assess how the model responds to extreme or boundary-case inputs.

  Use: Evaluates the model's performence under high-stress scenarios and find sensitivity to outliers, and checks whether predictions remain within reasonable bounds.

- **Test case 4 : Gender fairness test**

  Description: The gender feature is toggled while keeping all other features unchanged.

  Purpose: To check after changing only the gender leads to the various predictions.

  Use: This test developed to check the fairness and the bias of the model especially in sensitive applications, and ensures that the model does not produce discriminatory outputs based on gender information.

- **Test case 5 : Repetition of test**

  Description: The same input sample is repeated 10 times.

  Purpose: To test the predictions consistency.

  Use: Checks that predictions are remain stable.

Chapter 4

EXPERIMENTS AND RESULTS

Metric	Multiple Linear Regression	Decision Tree	Random Forest	SVR	XGBoost
MAE	2.8823	0.9347	0.7774	0.0036	0.3669
MSE	32.2348	4.2360	2.8005	0.0494	0.3400
RMSE	5.6776	2.0582	1.6735	0.2223	0.5831
R^2 Score	0.8351	0.9736	0.9745	0.9986	0.9966
Adjusted R^2	0.7362	0.9705	0.9712	0.9978	0.8869
MAPE (%)	24.0689	2.4002	1.7868	0.0012	0.8821
SMAPE (%)	8.3177	2.3125	1.7420	0.0014	0.1520
Median AE	1.6226	0.0200	0.4313	0.0012	0.0270
MBE	0.7569	0.5065	0.1062	-0.0036	0.9966
EVS	0.8380	0.5065	0.9746	0.9986	0.0452
RAE	0.2801	0.9752	0.1001	0.0494	0.0034
RSE	0.1649	0.0945	0.0255	0.0255	0.0179
AIC	341.3167	160.0308	110.6211	15.4118	-80.0181
BIC	413.1502	185.7779	135.3944	26.7108	-60.1091
Training Time (s)	6.1600	0.5006	78.2428	0.4764	476.3325
Prediction Time (s)	0.0000	0.0100	0.0334	0.0012	0.0147
Model Size (KB)	30.9043	30.9043	3501.0293	22.887	90.4912

Table 4.1: All models performance metrics

A detailed analysis of different regression models was done using a number of performance metrics to see how well they are working and the different models being looked at were Linear Regression, Decision Tree, Random Forest, Support Vector Regression (SVR) and XGBoost. We looked at these models error metrics, fit statistics, computational efficiency and model complexity.

As shown in Table 4.1, beginning with the Mean Absolute Error (MAE), SVR demonstrated the highest accuracy with the lowest error (0.0036), indicating that its predictions were remarkably close to the actual values. XGBoost also performence is well (0.3669), followed by Random Forest (0.7774) and decision Tree and Linear Regression exhibited

slightly greater errors (0.9347 and 2.8823, respectively), suggesting less precise predictions.

Moving on to the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), SVR again achieved the lowest values (MSE: 0.0494; RMSE: 0.2223), emphasizing its ability to minimize large deviations. XGBoost and Random Forest showed fairly good performance, while Decision Tree and Linear Regression reported higher squared errors, indicating their inferior robustness in handling this dataset. The R^2 value, which shows how well a model explains the differences in the data, also showed that SVR was better, with a value of 0.9986, which is almost ideal. XGBoost (0.9966) and Random Forest (0.9745) came in second and third while Decision Tree (0.9736) and Linear Regression (0.8351) came in last. The Adjusted R^2, which favors simpler models by applying a penalty to the complex ones, gave comparable findings. SVR came in to the lead, although XGBoost fell second, probably it is an ensemble model.

In terms of relative error metrics, the Mean Absolute Percentage Error (MAPE) and Symmetric MAPE (SMAPE) reaffirmed SVR's accuracy, with extremely low values (MAPE: 0.0012%; SMAPE: 0.0014%). XGBoost remained a strong contender here, whereas the other models, particularly Linear Regression, exhibited noticeably higher relative error percentages.When looking at Median Absolute Error (MedAE) and Mean Bias Error (MBE), SVR continued to stand out with minimal bias and consistent predictions. XGBoost and Random Forest also showed stable behavior, while Linear Regression had higher central errors and bias, implying over- or underestimation trends.

The Explained Variance Score (EVS) presented an interesting contrast—while SVR and Random Forest maintained high variance explanation capabilities, XGBoost's score dropped significantly, possibly due to specific data characteristics or noise sensitivity. Nonetheless, it remained strong across other metrics. Further, Relative Absolute Error (RAE) and Relative Squared Error (RSE) provided insight into how each model fared compared to a baseline predictor. XGBoost and SVR had the lowest error ratios, signifying strong generalization. Random Forest showed decent performance, but Decision Tree and Linear Regression were less reliable in this context.

In terms of model complexity, the Akaike Information Criterion (AIC) and Bayesian

Information Criterion (BIC) revealed that XGBoost, despite being more resource-intensive, provided an optimal trade-off between accuracy and complexity. SVR also have low AIC and BIC values while the Decision Tree and Linear Regression appeared over simplified and less effective under these criteria. From a computational standpoint, Linear Regression was the fastest to train, taking just 6.16 seconds. SVR and Decision Tree were also efficient in training. But Random Forest and XGBoost took a lot longer to train because they used ensemble methods. Even still all of the models had short prediction periods and SVR and Linear Regression gave findings almost right away, while Random Forest was a little later. Finally, in terms of model size SVR and Linear Regression were notably compact consuming under 35 KB. XGBoost occupied around 90 KB while Random Forest due to the large number of trees was considerably larger (around 3.5 MB) which may have implications for deployment in memory constrained environments.

In summary SVR has the most balanced model which will offering excellent predictive accuracy and low error rates, minimal bias, and fastest computation. XGBoost closely followed slightly outperforming in some complexity sensitive metrics but at the cost of training time and size of model. Random Forest was a dependable mid ground, while Decision Tree and Linear Regression served as faster but less accurate alternatives. This detailed analysis points to SVR and XGBoost as the top choices when accuracy and consistency are crucial for practical use.

4.1 Real time Test cases

4.1.1 Multiple linear regression with interaction terms

Model Test cases and interpretations

- **Test case 1: Real world batch**

  Input: The average same input sample is given to the model 1000 times at the time of training to check the stable real-world scenario.

  Explanation: The prediction is 46.00,the model gives a good prediction for data from the real world, so the model is stable.

- **Test case 2: All Inputs Set to Zero**

  Input: All Zero Input to test the model .

  Explanation: Model prediction is 1.76. The model is not giving a stable output even when all of the input features are zero. So this edge case failed.

- **Test case 3: All inputs set to high**

  Input: All high inputs to check the model consistency.

  Explanation: The prediction is -18.75. The model's output becomes negative when all the inputs are high, indicating that this model is sensitive to high input features.This Test case also failed.

- **Test case 4: Gender**

  Input: Only the gender feature is changing, and all the other values stay the same.

  Explanation: Both cases will be predicted at 32.00. The output remaining constant indicates that gender does not affect the prediction.

- **Test case 5: Repeat the same input**

  Input: Use the same input 10 times and check the output.

  Explanation: Predicted is 46.70 and 46.35 (ten times), The model gives little variation in results every time , which shows that it behaves in a deterministic way.

4.1.2 Updated decision tree

Model Test cases and interpretations

- **Test case 1: Real world batch**

  Input: The average same input sample is given to the model 1000 times at the time of training to check the stable real-world scenario.

  Explanation: The prediction is 46.00, indicating that the model provides a good estimate for real-world data. So this Test case passed.

- **Test case 2: All inputs set to low**

  Input: All zero input is to test the response about the model according to zero all input feature circumstance.

  Explanation: The model The prediction is 43.71. The model gives a stable output even when all of the input features are zero.

- **Test case 3: All inputs set to high** .

  Input: All high inputs set to high to check the model consistency.

  Explanation: The prediction is 73.00. The model's output increases when all the inputs are high, still it is in target output range, so this edge case passed.

- **Test case 4: Gender**

  Input: Only the gender feature is changing, and all the other values stay the same.
  Explanation: Both cases will be predicted at 47.13 and 48.04. The output remaining constant indicates that gender does not affect the prediction in this configuration.

- **Test case 5: Repeat the same input**

  Input: Use the same input 10 times and check the output.

  Explanation: The predicted value is 46.00, which remains consistent across ten trials, indicating that the model behaves in a deterministic manner.

4.1.3 Domain based random forest with staking

Model Test cases and results

- **Test case 1: Real world batch**

  *Input:*The average same input sample is given to the model 1000 times at the time of training to check the stable real-world scenario.

  Explanation: The prediction is 46.00. The model demonstrating strong stability.

- **Test case 2: All inputs set to low**

  Input: Set all zero input is to test the response about the model .

  Explanation: Prediction is 46.00. The model effectively manages edge cases without errors when all features are zero.

- **Test case 3: All inputs set to high**

  Input: All high inputs to check the model consistency.

  Explanation: Prediction is 46.00. The model is stable even with high input feature values represent robustness as well as no problem of overfitting.

- **Test case 4: Gender Fairness Check**

  Input: Only the gender feature is changing, and all the other values stay the same.

  Explanation: Prediction is 46.00 for both male and female and indicating that the model treats gender fairly and shows no bias .

- **Test case 5: Repeat same input**

  Input: The same valid input is passed to the model 10 times in a row.

  Explanation: Prediction is 46.00 for ten times, consistent predictions prove the model is deterministic.

4.1.4 SVR with granular bells

Model test cases and results

- **Test case 1: Real world batch**

  Input: The average same input sample is given to the model 1000 times at the time of training to check the stable real-world scenario.

  Explanation: Prediction is 45.11,The model gives a good prediction for data from the real world. So the model is stable.

- **Test case 2: All inputs set to low**

  Input: All zero input is to test the response of the model

  Explanation: The model The prediction is 47.00. The model giving a stable output even when all of the input features are zero. .

- **Test case 3: All inputs set to high**

  Input: All high inputs to check the model consistency

  Explanation: The prediction is 47.00. The model's is stable and robust this Test case passed.

- **Test case 4: Gender**

  Input : Only the gender feature is changing and all the other values remains the same

  Explanation: Both cases will have a prediction of 50.17 and 47.17. The output remaining constant indicates that gender does not affect the prediction in this configuration.

- **Test case 5: Repeat same input**

  Input: Use the same input for 10 times and check the output

  Explanation: Predicted is 49.19 (ten times),The model gives the same results every time , which shows that it behaves in a deterministic way.

4.1.5 XGBoost with grey wolf optimization

Model Test cases and Results

- **Test 1: Real world batch**

  Input: The average same input sample is given to the model 1000 times at the time of training to check the stable real-world scenario.

  Explanation: Prediction is 44.16,The model gives a good prediction for data from the real world. So the model is stable.

- **Test case 2: All inputs set to low**

  Input: All zero input is to test the response of the model

  *Explanation:*A successful prediction of 33.63. The model giving a stable output even when all of the input features are zero.

- **Test case 3: All inputs set to high**

  Input: All high inputs to check the model consistency.

  Explanation: the prediction is 33.24. The model's is stable and robust this Test case passed.

- **Test case 4: Gender**

  Input: Only the gender feature is changing and all the other values remains the same

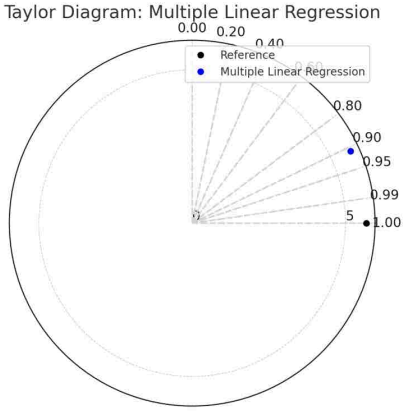
  Explanation: The prediction is (44.16 and 44.25) The output remaining constant indicates that gender does not affect the prediction in this configuration.

- **Test case 5: Repeat same input**

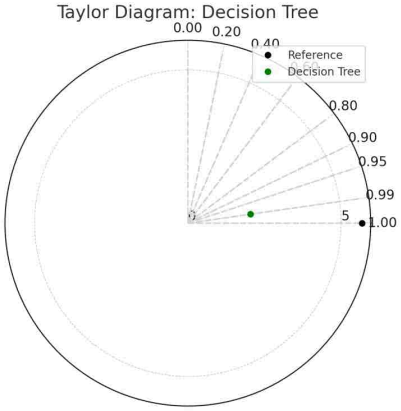
  Input: Use the same input for 10 times and check the output.

  Explanation: prediction is 45.11, (ten times),the model gives the same results every time , which shows that it behaves in a deterministic way.

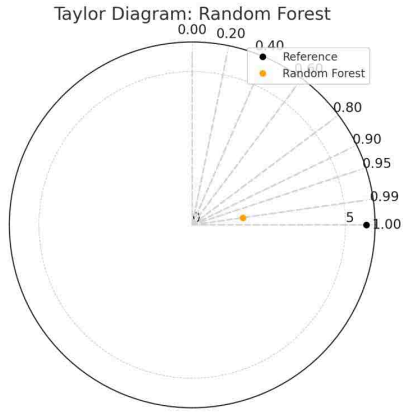
4.2 Graphs: Taylor plots



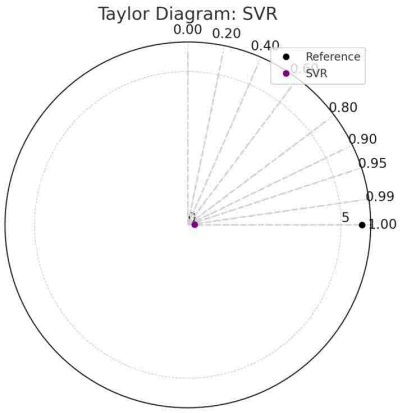
(a) Multiple Linear Regression



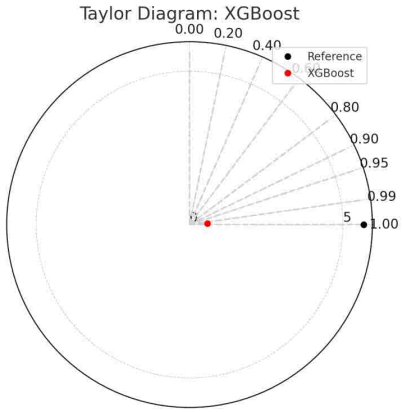
(b) Decision Tree



(c) Random Forest



(d) SVR



(e) XGBoost

Figure 4.1: Taylor diagrams for different models

Real vs. predicted plots are sometimes used to visually check how well a model works, but they may not always give a whole picture, especially when several models do well and their forecasts are quite near to the actual values. In certain instances, the visual distinctions across models may diminish, perhaps resulting in ambiguity or misinterpretation.

Smith and Rose (2008), in their paper published in Ecological Modelling [40], stressed that using only visual regression plots is not enough for a thorough model evaluation. These kinds of visual evaluations can hide variations in performance and make sure that the model is correct. So, techniques for quantitative and statistical validation, such as Taylor diagrams, residual analysis, and error metrics, are important for a more accurate and objective evaluation of predictive models. This is a comparison of the five models based only on their Taylor diagram representations. It looks at three main things: the correlation coefficient, the standard deviation, and the centered RMSE.

In the Taylor diagram, the correlation coefficient is indicated by the angle, the standard deviation by the radial distance from the origin, and the centered RMSE by the distance between the model point and the reference point.

The Taylor plot reveals that the Multiple Linear Regression (MLR) model is the least aligned. It is far from the reference point, has a poor correlation coefficient (0.8351), and its standard deviation does not reflect the variability that was seen. The model's high centered RMSE shows that it still has a lot of average deviation even when bias is removed. The diagram shows that MLR doesn't work well for capturing the target's complexity, especially in areas that aren't linear.

The Decision Tree model is better because it has a higher correlation (0.9736) and a standard deviation that is closer to the mean than MLR. The Taylor diagram shows that it is much closer to the reference point, which means it is more reliable. But it still has a little higher angular spread, which means that there is still some error and variance that isn't entirely matched with the genuine distribution. This is in line with its tendency to overfit and make predictions that aren't continuous.

The Random Forest model is quite strong on the Taylor plot, with a very high correlation (0.9745) and a standard deviation that is almost the same as that of the reference. The fact that it is relatively close to the reference point means that the centered RMSE is very

low, which means that the model's predictions always match the observed values in both size and direction. This strong alignment shows how well the ensemble model can capture complicated, non-linear interactions while avoiding overfitting by averaging.

The Support Vector Regression (SVR) model fits almost perfectly on the Taylor diagram. It has the highest correlation (0.9986) and a standard deviation that is almost the same as the data that was seen. The centered RMSE is the least, which means that SVR is almost on top of the reference point. This picture shows that SVR not only makes very accurate predictions, but it also captures the variability of the true values with very little inaccuracy. When looking at the Taylor plot, SVR is the most statistically accurate and consistent model.

XGBoost is also extremely close to the reference point, however it is a little behind SVR. It has a correlation of 0.9966, which means it agrees with observed values quite well, and its standard deviation is also very close. The Taylor diagram shifts slightly because the centered RMSE is a little higher than the SVR. Even so, XGBoost's position shows that it is a high-quality model that can make accurate and steady predictions, especially when working with complicated datasets.

Finally, the Taylor diagram confirms that SVR and XGBoost are the most statistically accurate models for this prediction task, while Random Forest remains a solid alternative. Models like Decision Tree and MLR lack the precision and consistency required for high-stakes applications like clinical decision-making.

Model Performance Graphs

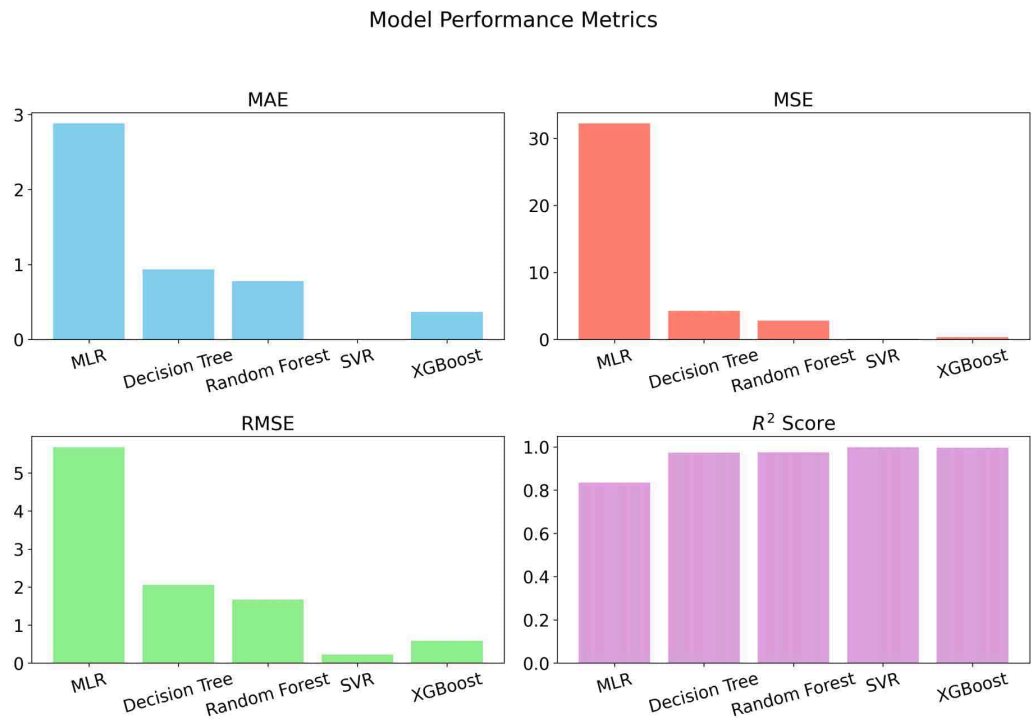


Figure 4.2: SVR model performance for different number of features

To evaluate the performance of various regression models here we used four standard metrics as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and the R² Score. These metrics help to know how much accurate and reliable each model performs.

The results illustrated in Fig. 4.2, demonstrate In case of MAE, SVR and XGBoost got the lowest values, which means their predictions was very close to the actual output on average. On the other hand, Multiple Linear Regression showed highest MAE, so its prediction are more off.

The MSE results also shows a similar pattern. Since MSE penalize large errors more, the small MSE values from SVR and XGBoost indicates that they didn't make big mistakes. Again, Multiple Linear Regression performed worst with largest MSE.

RMSE gives error in original units of target variable. SVR and XGBoost done well here too, giving low RMSE values. Random Forest and Decision Tree did decent, but not that great. Multiple Linear Regression showed high RMSE, which is not much good.

R^2 Score tells how much variation in target is explained by model. SVR had the highest R^2 around 0.9866, and XGBoost also had a good R^2 after SVR. But Multiple Linear Regression got only 0.8353 so it not able to explain much variance in data.

So overall, SVR and XGBoost are the best model among the tested ones. They gave better accuracy and consistency. Multiple Linear Regression was not suitable for this problem due to its poor results in all metrics.

Impact of Feature Selection on Model Performance

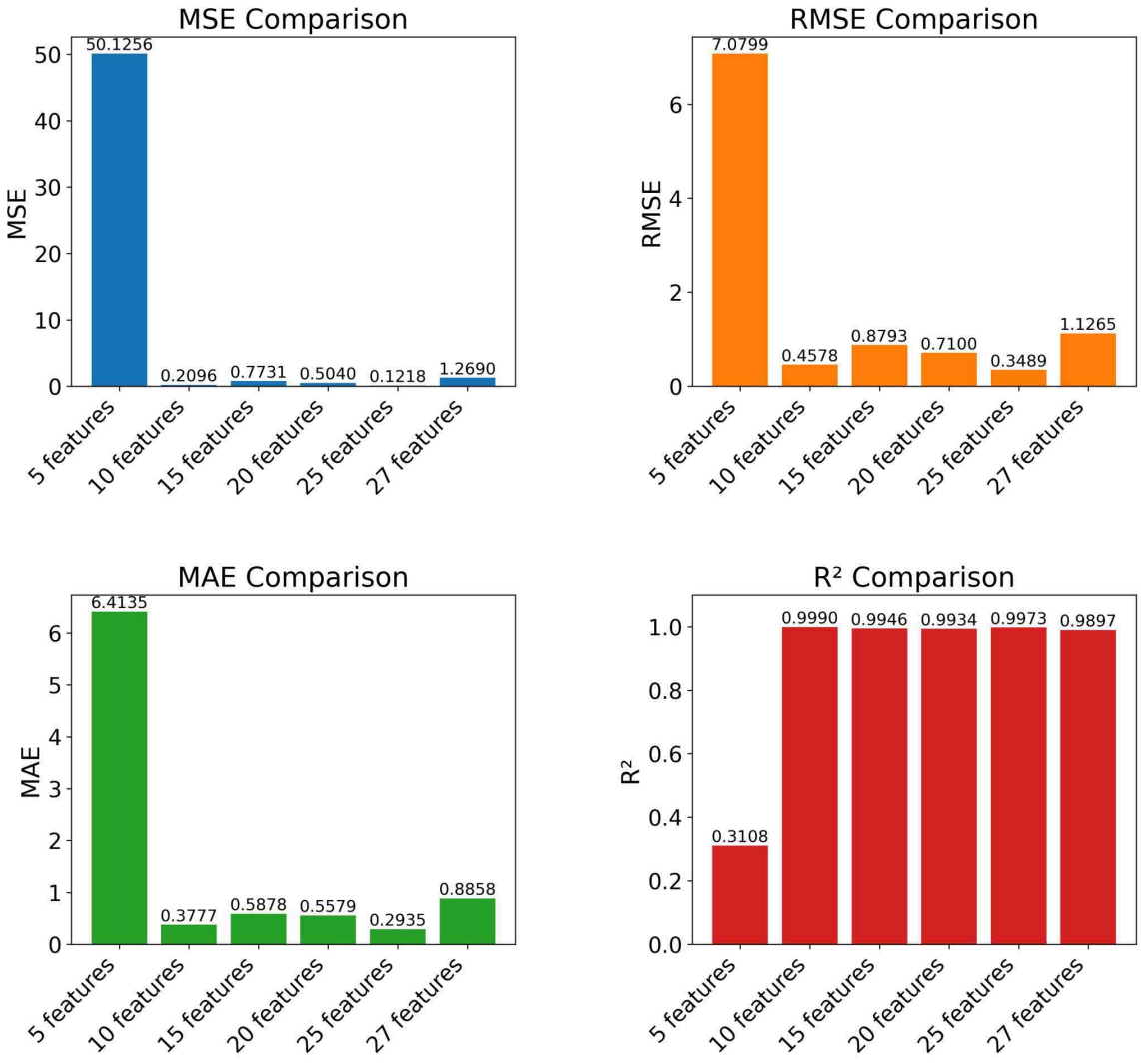


Figure 4.3: SVR model performance for different number of features

Tested a robust model called Improved SVR using granular bells with several combi-

nations of input features, ranging from 5 to 30, to see how well it could predict dry weight in hemodialysis patients. The purpose was to see how the number of clinical features influences how well the model predicts. Some of the key approaches to verify how accurate and dependable the model were Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). These measures show us alternative ways to think about prediction error and which helps us figure out how the model works when the input gets more intricate.

The results illustrated in Fig. 4.3, demonstrate When tested with only five features, the model had a high error rate (MSE of 50.13, RMSE of 7.08, and MAE of 6.41) and a low R^2 value (0.3108) these results show that the model doesn't fit well and that it did not capture sufficient variation. This kind of performance shows that the model isn't fitting well, which means it doesn't have enough data to figure out the patterns that control fluid balance in persons who are on hemodialysis.

Adding 10 more features made a big difference in performance. The model had a very low MSE of 0.2096, RMSE of 0.4578, and MAE of 0.3777, and its R^2 value was almost perfect at 0.9990. These results show that adding more relevant clinical characteristics, such gender, blood pressure, and temperature, makes the model much better at generalizing. It indicates that these variables encompass critical information regarding the patient's fluid status and physiological responses to dialysis.

The model still worked well when the number of features was raised to 15 and 20 with the R^2 values of 0.9946 and 0.9934, respectively. But the MAE and RMSE did go up a little, which suggests that the returns are getting smaller. For example MAE went up to 0.5878 and RMSE went up to 0.8793 when there were 15 features. This means that some more features helped keep accuracy while others may have added noise or redundancy which caused the model's performance to drop slightly. So, these results show that we are at a point where adding more complexity does not lead to commensurate advantages.

At 25 features, the model found the best mix between performance and complexity. The MSE went down to 0.1218, the RMSE went down to 0.3489, and the MAE went down to 0.2935. The R^2 value was very good at 0.9973. This means that this set of features gave the highest generalizable clinical signal for predicting dry weight. It probably includes a

full set of dialysis parameters, vital signs, and treatment information. This result indicates that 25 characteristics may constitute the optimal threshold for practical clinical models: adequate information with little overfitting risk.

Adding more features (30 in all) made performance worse, which is interesting. The MSE went up to 1.269, the RMSE went up to 1.1265, and the MAE went up to 0.8858. The R^2 went down to 0.9897. This drop suggests overfitting, which happens when the model gets too complicated and starts to memorize the training data instead of learning general patterns. Adding features that weren't useful or that were too similar to each other probably confused the model, which made it fit noise. This result shows how important it is to choose the right features and how dangerous it is to blindly make the model more complex. Finally, the analysis demonstrates that the judicious addition of beneficial characteristics significantly enhances predictive performance, whereas the introduction of unnecessary complexity detracts from it. The model seems to fare best at balancing clinical relevance with predicting accuracy when there are between 10 and 25 characteristics. These findings are crucial for the development of decision-support systems in dialysis care, as accurate dry weight estimation can directly impact patient outcomes by preventing overhydration or dehydration.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis provided Meaningful insights into the application of different machine learning models for predicting Dry Weight in dialysis patients. We employed five models Multiple Linear Regression, Decision Tree, Random Forest, Support Vector Regressor (SVR), and XGBoost. We carefully tested these models with a number of performance metrics.

The results showed that SVR model had the best match and the less error rate, but because it takes longer to train, It can be challenging to apply to large scaled datasets. XGBoost, on the other hand, was a strong choice for real-time clinical usage since it offered a good balance of accuracy, speed, and model size. The Random Forest and Decision Tree models do okay, but they are easier to understand and train faster, which could be helpful in situations where model simplicity is more important. Multiple Linear Regression, however the least precise, provided valuable interpretative insights into the linear associations between features and the target variable.In general, it is very important to choose the right models for healthcare applications depending on how accurate, how much processing power, and how easy they are to understand they need to be.

5.2 Future Work

This work can be made better and more useful in the future. This project only has 517 records, hence the dataset is limited. The model will be more accurate and function for more patients if we can gather more data from other hospitals. A larger dataset will also help with the overfitting problem.

We also employed regular machine learning models on data that was in tables. We can employ deep learning models like LSTM, GRU, or Transformer models in the future. These models work well with data that changes over time. If we have more than one session with the same patient, we may also guess how their Dry Weight will fluctuate over time.

Additionally, the model's predictive capabilities might be enhanced by including other input variables. This might contain complete blood test results, diaries of food and drink intake, medication schedules and dosages, and other important clinical information. Adding these characteristics would provide a complete picture of the patient's health and help the model find subtle trends, which would make it more accurate and useful in a therapeutic setting.

Bibliography

[1] J. E. Flythe, S. E. Kimmel, and S. M. Brunelli, "Rapid fluid removal during dialysis is associated with cardiovascular morbidity and mortality," *Kidney Int.*, vol. 79, no. 2, pp. 250–257, 2014.

[2] C. Chazot, B. Charra, C. Vo Van, G. Jean, T. Vanel, E. Calemard, and G. Laurent, "The importance of dry weight in hemodialysis," *Nephrol. Dial. Transplant.*, vol. 14, no. 1, pp. 121–124, 1999.

[3] A. D. Sinha, R. P. Light, and R. Agarwal, "Relative plasma volume monitoring during hemodialysis aids the assessment of dry weight," *Hypertension*, vol. 60, no. 2, pp. 308–314, 2013.

[4] M. Onofriescu et al., "Bioimpedance-guided fluid management in maintenance hemodialysis: A pilot randomized controlled trial," *Am. J. Kidney Dis.*, vol. 64, no. 1, pp. 111–118, 2014.

[5] F. Mallamaci et al., "Detection of pulmonary congestion by chest ultrasound in dialysis patients," *J. Am. Coll. Cardiol.*, vol. 53, no. 7, pp. 668–673, 2010.

[6] R. Agarwal and M. R. Weir, "Dry-weight: A concept revisited in an effort to avoid medication-directed approaches for blood pressure control in hemodialysis patients," *Clin. J. Am. Soc. Nephrol.*, vol. 5, no. 7, pp. 1255–1260, 2010.

[7] J. Kim et al., "Machine learning approaches for fluid imbalance detection in hemodialysis patients," *Artif. Intell. Med.*, vol. 102, pp. 101754, 2020.

[8] J. P. Kooman, F. M. van der Sande, and K. M. Leunissen, "Continuous blood pressure monitoring in hemodialysis: The role of wearable technology," *Nat. Rev. Nephrol.*, vol. 16, no. 1, pp. 11–12, 2020.

[9] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, pp. 199–222, 2004.

[10] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. Hoboken, NJ, USA: Wiley, 2021.

[11] A. Liaw and M. Wiener, "Classification and regression by RandomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[12] G. R. Lakshminarayanan et al., "Hemodialysis outcomes and practice patterns in end-stage renal disease: Experience from a Tertiary Care Hospital in Kerala," *Indian J. Nephrol.*, vol. 27, no. 1, pp. 51–57, 2017.

[13] R. Agarwal and M. R. Weir, "Dry-weight: a concept revisited in an effort to avoid medication-directed approaches for blood pressure control in hemodialysis patients," *Clin. J. Am. Soc. Nephrol.*, vol. 5, no. 7, pp. 1255–1260, 2010.

[14] A. Shrestinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D*, vol. 404, 2020.

[15] T. Kimura et al., "Deep learning analysis of clinical course of primary nephrotic syndrome: Japan Nephrotic Syndrome Cohort Study (JNSCS)," *Clin. Exp. Nephrol.*, vol. 26, no. 12, 2022.

[16] O. Niel and P. Bastard, "Artificial intelligence in nephrology: core concepts, clinical applications, and perspectives," *Am. J. Kidney Dis.*, vol. 74, no. 6, 2019.

[17] A. Scala et al., "Assessment of proteinuria level in nephrology patients using a machine learning approach," in *Proc. 5th Int. Conf. Med. Health Inform.*, 2021.

[18] B. J. Erickson et al., "Machine learning for medical imaging," *Radiographics*, vol. 37, no. 2, 2017.

[19] A. Nayyar, L. Gadhavi, and N. Zaman, "Machine learning in healthcare: review, opportunities and challenges," in *Proc. Mach. Learn. Internet Med. Things Healthc.*, 2021.

[20] C. Basile et al., "Development and validation of bioimpedance analysis prediction equations for dry weight in hemodialysis patients," *Clin. J. Am. Soc. Nephrol.*, vol. 2, no. 4, 2007.

[21] A. K. Singh, P. Sharma, and A. Chellapan, "Predicting Dry Weight in Hemodialysis Patients Using Deep Learning Techniques," *arXiv preprint arXiv:2401.00761*, 2024.

[22] E. Nobakht et al., "Precision Dialysis: Leveraging Big Data and Artificial Intelligence," *Kidney Med.*, vol. 6, no. 9, pp. 100868, 2024.

[23] Y. Gorishniy et al., "Revisiting Deep Learning Models for Tabular Data," *arXiv preprint arXiv:2106.11959*, 2021.

[24] C. Ai and E. C. Norton, "Interaction Terms in Nonlinear Models," *Health Serv. Res.*, Aug. 2011.

[25] C. Bertsimas and J. Dunn, "Regularized impurity reduction: Accurate decision trees with complexity guarantees," *Data Min. Knowl. Discov.*, vol. 36, no. 4, pp. 1155–1192, 2022.

[26] C. F. Kurz, W. Maier, and C. Rink, "A greedy stacking algorithm for model ensembling and domain weighting," *BMC Res. Notes*, vol. 13, no. 70, 2020.

[27] Y. Li et al., "Construction of a predictive model based on MIV-SVR for prognosis and length of hospital stay in traumatic brain injury," *Comput. Math. Methods Med.*, vol. 2023, Article ID 6683342, 2023.

[28] R. U. Rahman et al., "A novel machine learning-based artificial intelligence approach for log analysis using blockchain technology," *Sigma J. Eng. Nat. Sci.*, vol. 42, no. 5, pp. 1391–1409, Oct. 2024.

[29] M. Wu et al., “Beyond Sparsity: Tree Regularization of Deep Models for Interpretability,” in Proc. AAAI Conf. Artificial Intelligence, vol.32, Feb.2018. doi: 10.1609/aaai.v32i1.11501.

[30] H. W. Kim et al., ”Dialysis adequacy predictions using a machine learning method,” *Sci. Rep.*, vol. 11, no. 1, pp. 15417, 2021.

[31] M. E. Brier and A. E. Gaweda, ”Artificial intelligence for optimal anemia management in end-stage renal disease,” *Kidney Int.*, vol. 90, no. 2, pp. 259–261, 2016.

[32] C. Barbieri et al., ”An international observational study suggests that artificial intelligence for clinical decision support optimizes anemia management in hemodialysis patients,” *Kidney Int.*, vol. 90, no. 2, pp. 422–429, 2016.

[33] D. Cheng et al., ”GB-DBSCAN: A fast granular-ball based DBSCAN clustering algorithm,” *Inf. Sci.*, vol. 674, pp. 120731, 2024.

[34] S. Mirjalili, S. M. Mirjalili, and A. Lewis, ”Grey wolf optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[35] A. Valdivia, J. Sánchez-Monedero, and J. Casillas, ”How Fair Can We Go in Machine Learning, Assessing the Boundaries of Fairness in Decision Trees,” *arXiv preprint arXiv:2006.12399*, 2020.

[36] H. Koul, B. Altmann, and J. Rahnenführer, ”Knowledge-slanted random forest for biologically relevant feature selection,” *BioData Min.*, vol. 17, no. 1, pp. 1–17, 2024.

[37] Y. Zhang et al., ”Clinical data-based XGBoost algorithm for infection risk prediction of decompensated cirrhosis patients,” *BMC Gastroenterol.*, vol. 23, no. 1, pp. 1–11, 2023.

[38] R. Agarwal and M. R. Weir, “Dry-weight: A concept revisited in an effort to avoid medication-directed approaches for blood pressure control in hemodialysis patients,” Clin. J. Am. Soc. Nephrol., vol. 5, no. 7, pp. 1255–1260, Jul. 2010.

[39] A. Kamijo, Y. Oda, T. Hasegawa, S. Fukui, T. Ohashi, T. Iwata, and K. Hirayama, "Predicting dry weight change in hemodialysis patients using machine learning," BMC Nephrol., vol. 24, no. 1, pp. 1–9, Jul. 2023.

[40] G. Piñeiro, S. Perelman, J. P. Guerschman, and J. M. Paruelo, "How to evaluate models: Observed vs. predicted or predicted vs. observed?," Ecological Modelling, vol.216, no.3–4, pp.316–322, Sep. 2008, doi:10.1016/j.ecolmodel.2008.05.006. :